

Randomized LU Decomposition Using Sparse Projections

Yariv Aizenbud¹ Gil Shabat² Amir Averbuch²

¹School of Applied Mathematics, Tel Aviv University, Israel

²School of Computer Science, Tel Aviv University, Israel

January 19, 2016

Abstract

A fast algorithm for the approximation of a low rank LU decomposition is presented. In order to achieve a low complexity, the algorithm uses sparse random projections combined with FFT-based random projections. The asymptotic approximation error of the algorithm is analyzed and a theoretical error bound is presented. Finally, numerical examples illustrate that for a similar approximation error, the sparse LU algorithm is faster than recent state-of-the-art methods. The algorithm is completely parallelizable that enables to run on a GPU. The performance is tested on a GPU card, showing a significant improvement in the running time in comparison to sequential execution.

Keywords. LU decomposition, random matrices, sparse matrices, sparse Johnson-Lindenstrauss transform.

1 Introduction

Low-rank matrix approximations are a key component for efficient processing, manipulating and analysis of big datasets. Often, data matrices can be very large and yet have many redundancies and dependencies between rows and columns that result in being a low-rank matrix. Finding a low-rank approximation of a matrix enables us to process the entire matrix by using only a small set of vectors. Applications that utilize low-rank matrix approximations include data compression, noise filtering, principle component analysis and kernel methods, to name some. Although a low-rank matrix approximation can be computed using well-known matrix decomposition methods, such as singular value decomposition (SVD) or rank revealing QR (RRQR), very often this is impractical due to high computational load. Therefore, there is an ongoing interest in the development of fast algorithms for computing low-rank matrix approximations. Randomized algorithms for low rank matrix approximations include SVD [8, 10, 18], LU [15], CUR [4, 5], principal component analysis (PCA) [7, 16], to name some. Randomized algorithms have gained an increasing popularity because of their abilities to perform matrix computations faster and on larger data sets than classical algorithms such as [6].

Sparse random projections have been studied for dimensionality reduction as a sparse variant of the Johnson-Lindenstrauss (JL) transform. A nearly tight lower bound for several dimensionality reduction linear maps for a predetermined sparsity is given in [12, 13].

Algorithms, which utilize sparse random projections for SVD and regression computations, are given in [1, 3, 9, 11]. Algorithms that are based on sparse dimensionality reduction transforms benefit from the fact that their projection step is more computationally efficient than those that use dense matrices in their projection step. While the complexity of the algorithms, which use a structured JL transform such as FFT-based random projections [18], does not change when applied to sparse matrices, algorithms that are based on sparse random projections are accelerated when applied to sparse matrices.

In this paper, the randomized LU algorithms [15] are extended by utilizing sparse random projections. We introduce an LU decomposition algorithm that uses sparse random projections combined with the fast Johnson-Lindenstrauss (FJL) transform. FJL transforms are based on the fast Fourier

transform (FFT) [2] and are also used in [18]. This combination of sparse JL with FJL was introduced in [3] to produce faster algorithms. The algorithm presented in this paper is shown to be significantly faster for a low-rank matrix decomposition than the algorithms mentioned above. In addition, a detailed theoretical analysis is presented for the derived error bounds of the algorithm.

For a given matrix A of size $m \times n$, the algorithm computes the lower and upper triangular matrices L and U of sizes $m \times k$ and $k \times n$, respectively, and permutation matrices P and Q such that with high probability

$$\|LU - PAQ\|_F \leq \mathcal{O}(\Delta_r) \quad (1)$$

where $\Delta_r \triangleq \sqrt{\sum_{i=r+1}^{\min(m,n)} \sigma_i^2}$, $r < k$. Then, the performance of the algorithm is compared with the current state-of-the-art methods that compute low-rank matrix approximations. The presented algorithm is parallelizable and can be fully implemented on a GPU.

The paper is organized as follows: Section 2 reviews some mathematical results that are needed for the development of the sparse randomized LU algorithm. Section 3 presents the sparse randomized LU algorithm and the error bound resulted from the approximation. Section 4 presents numerical results for the approximation error and for the running time of the sparse randomized LU with comparison to other algorithms.

2 Preliminaries

This section presents the mathematical background needed in the rest of the paper. More specifically, we review the properties of the Sub-sampled Random Fourier Transform (SRFT) matrices and the sparse embedding matrices. Throughout the paper, $\|\cdot\|_F$ denotes the Frobenius norm, $\|\cdot\|_2$ denotes the spectral norm when the argument is a matrix or the l_2 (Euclidean) norm for vector arguments. $M_{m \times n}$ is the set of $m \times n$ matrices, $\sigma_i(\cdot)$ is the i th largest singular value of a matrix, and $\Delta_k(\cdot) = \sqrt{\sum_{i=k+1}^{\min(m,n)} \sigma_i^2}$, $k = 0, \dots, \min(m, n) - 1$.

2.1 The SRFT matrix

The SRFT matrix, which is presented in [2, 18], is a random matrix denoted by Π . It is decomposed into $\Pi = DFS$ where D is an $n \times n$ diagonal matrix whose entries are i.i.d. random variables drawn from a uniform distribution on the unit circle in \mathbb{C} , F is an $n \times n$ discrete Fourier transform such that $F_{jk} = \frac{1}{\sqrt{n}} e^{-2\pi i(j-1)(k-1)/n}$, $j, k = 1, \dots, n$ and S is an $n \times l$ matrix whose entries are all zeros except for a single randomly placed 1 in each column.

Lemma 2.1 shows that matrix multiplication by an SRFT matrix can be done faster in comparison to an arbitrary matrix.

Lemma 2.1 ([18]). *For any $m \times n$ matrix A , let Π be the $n \times l$ SRFT matrix. Then, $Y = A\Pi$ can be computed in $\mathcal{O}(mn \log l)$ floating point operations.*

Theorem 2.2 (Follows from Theorem 1.3 in [17]). *For any $U \in M_{n \times r}$ with orthogonal columns, if $\Pi \in M_{k \times n}$, is a randomly chosen SRFT matrix, where r, k and n satisfy $4 \left[\sqrt{r} + \sqrt{8 \log(rn)} \right]^2 \log r \leq k \leq n$. Then, with probability of at least $1 - \mathcal{O}(r^{-1})$, the largest and the smallest singular values of ΠU are in $[0.40, 1.48]$.*

2.2 Sparse Embedding Matrices

For a parameter $t \in \mathbb{N}$, consider the random linear map $S = \Phi D$, where $S \in M_{k \times n}$, such that for $h : \{1, \dots, n\} \rightarrow \{1, \dots, k\}$, a random map such that for each $i \in \{1, \dots, n\}$, $h(i) = t'$ for $t' \in \{1, \dots, k\}$ with probability $1/t$, we have

1. $\Phi \in \{0, 1\}^{k \times n}$ is a $k \times n$ ($k \leq n$) binary matrix with nonzero entries $\Phi_{h(i), i} = 1$ and all the remaining entries equal to 0. In other words, Φ is a matrix with a single 1 in each row.
2. D is an $n \times n$ random diagonal matrix where each diagonal entry is independently chosen to be +1 or -1 with equal probability.

A matrix S that satisfies 1 & 2 is referred to as a sparse embedding matrix (SEM).

Lemma 2.3. Let $S \in M_{k \times n}$ be an SEM matrix. Then, $\|S\|_F = \sqrt{n}$.

Theorem 2.4. The largest singular value of a $k \times n$ SEM is bounded, with high probability, by $C(n, k) = \sqrt{\frac{n}{k}} + \sqrt{2\frac{n}{k} \log k}$ for large enough n .

The proof of Theorem 2.4 uses Lemma 2.5:

Lemma 2.5. The operator norm of an SEM $S \in M_{k \times n}$ is the square root of the maximal number of non-zeros in a row in S .

Proof. Assume, without loss of generality, that there are κ_i , $i = 1, \dots, k$ non-zeros in each row, $\kappa_1 \geq \dots \geq \kappa_k$. Denote the set of non-zero indices in the i th row by K_i ($|K_i| = \kappa_i$, $i = 1, \dots, k$). Since there is only one non-zero in each column, $\sum_{i=1}^k \kappa_i = n$. There is a vector v of unit length such that $\|Sv\| = \sqrt{\kappa_1}$. Let $v = (v_1, \dots, v_n)^T$ be such that $\sum_{i=1}^k v_i^2 = 1$. Then

$$\|Sv\|_2^2 = \sum_{j=1}^k \left(\sum_{i \in K_j} v_i \right)^2.$$

Since $\max_{\sum_{i \in K_j} v_i^2 = \alpha_j} \sum v_i$ is achieved when $v_i = \sqrt{\frac{\alpha_j}{\kappa_j}}$ for all $i \in K_j$, then we have

$$\|Sv\|^2 \leq \sum_{j=1}^k \left(\kappa_j \sqrt{\frac{\alpha_j}{\kappa_j}} \right)^2 = \sum_{j=1}^k (\sqrt{\kappa_j \alpha_j})^2 = \sum_{j=1}^k \kappa_j \alpha_j.$$

Since $\sum \alpha_j = 1$ it follows that $\|Sv\|^2 \leq \kappa_1$. Thus, $\|Sv\| \leq \sqrt{\kappa_1}$. \square

Remark 2.6. In a similar way, one can show that all the singular values of S are of the form $\sqrt{\kappa_i}$.

Proof of Theorem 2.4. By Lemma 2.5, the norm of S is the square root of the maximal number of non-zeros (nnz) in a row. The maximal nnz in each row is distributed as the maximum of n balls thrown into k urns. By Theorem 1 in [14], the probability of the norm to be more than $\frac{n}{k} + \sqrt{2\frac{n}{k} \log k}$ is $o(1)$. Thus, the norm is bounded, with high probability, by $\sqrt{\frac{n}{k}} + \sqrt{2\frac{n}{k} \log k}$ for sufficiently large n . \square

Theorem 2.7 (Appears as Theorem 3 in [11]). For any $U \in M_{m \times r}$ with orthogonal columns, if $S \in M_{l \times m}$ where $l \geq \delta^{-1}(r^2 + r)/(2\varepsilon - \varepsilon^2)^2$ is a randomly chosen SEM, then with probability of at least $1 - \delta$, the largest and smallest singular values of SU are in the interval $[1 - \varepsilon, 1 + \varepsilon]$.

Corollary 2.8. Let $\Omega = \Pi S$, where $\Pi \in M_{k \times l}$ is as in Theorem 2.2 and $S \in M_{l \times m}$ as in Theorem 2.7. Then, for any $U \in M_{m \times r}$, which has orthogonal columns with high probability, $\|\Omega U\|_2 \leq 1.48(1 + \varepsilon)$ and $\|(\Omega U)^{-1}\|_2 \leq 0.4 \frac{1}{(1 - \varepsilon)}$.

Theorem 2.9 (Appears as Lemma 46 in [3]). Let $A \in M_{m \times d}$ be of rank r , $B \in M_{m \times d'}$, and $c = d + d'$. For SEM $S \in M_{l \times m}$ and SRFT matrix $\Pi \in M_{k \times l}$, there exist $l = O(r^2 \log^6(r/\varepsilon) + r\varepsilon^{-1})$ and $k = O(r\varepsilon^{-1} \log(r/\varepsilon))$ such that for $\Omega = \Pi S$, $\tilde{X} = \operatorname{argmin}_X \|\Omega(AX - B)\|_F$ satisfies $\|A\tilde{X} - B\|_F \leq (1 + \varepsilon) \min_X \|AX - B\|_F$ with a fixed non-zero probability. The operator Ω can be applied in $O(\operatorname{nnz}(A) + \operatorname{nnz}(B) + lc \log l)$ operations.

An improved bound appears in [11] and is shown to be near optimal in [12].

3 Sparse Randomized LU

Similarly to the work presented in [8, 10], the key idea in the current algorithm is that the image of AS for a randomly chosen SEM S is “close” to the image of A up to an error of order Δ_r . It is shown in [8] that for each r there is $k > r$ such that if S is a random matrix of size $n \times k$ generated from the set of Gaussian i.i.d. matrices, or from SRFT matrices, then with high probability the image of AS is close to the image of A . More rigorously, if we denote by Q an $n \times k$ matrix with orthonormal columns that has the same image as AS , which is calculated by the QR algorithm, then $\|A - QQ^*A\|_F \leq \mathcal{O}(\Delta_r)$. We show in Theorem 3.1 that this is also true for the set of random SEM:

Theorem 3.1. Let A be an $m \times n$ matrix. Assume that $l = O(r^2 \log^6(r/\varepsilon) + r\varepsilon^{-1})$, $k = O(r\varepsilon^{-1} \log(r/\varepsilon))$, $\Pi \in M_{k \times l}$ is an SRFT matrix and $S \in M_{l \times n}$ is an SEM. Let $\Omega = \Pi S$ and the QR decomposition of $A\Omega^*$ is denoted by QR . Then, $\|A - QQ^*A\|_F < (1 + \varepsilon)\Delta_r(A)$.

The proof Theorem 3.1 uses ideas similar to some in [3].

Proof. First, we show that $\min_{\text{rank } X=r} \|QX - A\|_F \leq (1 + \varepsilon)\Delta_r$. Assume A_r is the best rank r approximation of A . Then, directly from this assumption, it follows that $\min_Y \|YA_r - A\|_F = \|A_r - A\|_F = \Delta_r$. From Theorem 2.9 follows that if $\tilde{Y} = \text{argmin}_Y \|(YA_r - A)\Omega^*\|_F$, then

$$\|\tilde{Y}A_r - A\|_F \leq (1 + \varepsilon) \min_Y \|YA_r - A\|_F = (1 + \varepsilon)\Delta_r.$$

Note that

$$\text{argmin}_Y \|(YA_r - A)\Omega^*\|_F = \text{argmin}_Y \|YA_r\Omega^* - A\Omega^*\|_F = A\Omega^*(A_r\Omega^*)^\dagger.$$

Thus,

$$\|A\Omega^*(A_r\Omega^*)^\dagger A_r - A\|_F \leq (1 + \varepsilon)\Delta_r. \quad (2)$$

From Eq. (2) it follows that

$$\min_{\text{rank } X=r} \|A\Omega^*X - A\|_F \leq (1 + \varepsilon)\Delta_r, \text{ where } X \in M_{k \times n}.$$

By using the fact that

$$\min_{X \text{ s.t. rank } X=r} \|QX - A\|_F \leq \min_{X \text{ s.t. rank } X=r} \|A\Omega^*X - A\|_F$$

we get

$$\min_{X \text{ s.t. rank } X=r} \|QX - A\|_F \leq (1 + \varepsilon)\Delta_r. \quad (3)$$

It follows that $\|QQ^*A - A\|_F \leq \min_{X \text{ s.t. rank } X=r} \|QX - A\|_F$, which concludes the proof. \square

Theorem 3.1 shows that QQ^*A approximates A well. Since Q and Q^*A are relatively small matrices and since Q has orthogonal columns, then the SVD computation of Q^*A is faster than the SVD computation of A . Unfortunately, Q^* is a dense matrix, then the multiplication Q^*A is computationally expensive. We now show how to replace the computation of Q^*A with a multiplication of A by a sparse matrix without affecting the accuracy too much.

Corollary 3.2. Let A be a $m \times n$ matrix. Assume $l = O(r^2 \log^6(r/\varepsilon) + r\varepsilon^{-1})$, $k = O(r\varepsilon^{-1} \log(r/\varepsilon))$, $\Pi \in M_{k \times l}$ is an SRFT matrix and an SEM $S \in M_{l \times n}$. Denote $\Omega = \Pi S$ and the pivoted LU decomposition of $A\Omega^*$ is denoted by $PA\Omega^* = LU$. Then $\|PA - LL^\dagger PA\|_F < (1 + \varepsilon)\Delta_r(A)$.

Proof. The proof is the same as that of Theorem 3.1. The reason that the same proof works is that $\text{Im } L = \text{Im } Q$. \square

Algorithm 3.1: Sparse Randomized LU Decomposition

Input: A matrix of size $m \times n$ to decompose; approximation rank $r < n$; $k_1 < l_1 < k_2 < l_2$ number of columns to use in the projections and the size of output matrices.

Output: Matrices P, Q, L, U such that $\|PAQ - LU\|_F \leq \mathcal{O}(\Delta_r(A))$, where P and Q are orthogonal permutation matrices, L and U are lower and upper triangular matrices, respectively.

- 1: Create a random SEM $S_1 \in M_{l_1 \times n}$ and an SRFT matrix $\Pi_1 \in M_{k_1 \times l_1}$. Let $\Omega_1 = \Pi_1 S_1$ be of size $k_1 \times n$.
 - 2: Compute $B = A\Omega_1^*$ ($B \in M_{m \times k_1}$).
 - 3: Compute the LU decomposition of B : $PB = L_1 U_1$, where $L_1 \in M_{m \times k_1}$ is a lower triangular matrix and $U_1 \in M_{k_1 \times k_1}$ is an upper triangular matrix.
 - 4: Create a random SEM S_2 of size $l_2 \times m$ and an SRFT matrix $\Pi_2 \in M_{k_2 \times l_2}$. Let $\Omega_2 = \Pi_2 S_2$ be of size $k_2 \times m$.
 - 5: Compute $\Omega_2 L_1$ and $(\Omega_2 L_1)^\dagger$.
 - 6: Compute the LU decomposition with right partial pivoting of $(\Omega_2 L_1)^\dagger \Omega_2 PA$ such that $(\Omega_2 L_1)^\dagger \Omega_2 PAQ = \tilde{L}U$.
 - 7: $L \leftarrow L_1 \tilde{L}$.
 - 8: Return L, U, P, Q
-

Theorem 3.3 (Correctness of the algorithm). *Let A be an $m \times n$ matrix. The sparse randomized LU decomposition of A uses the integers $k_1 = \mathcal{O}(r \log(r))$, $k_2 = \mathcal{O}(r)$, $l_1 = \mathcal{O}(r^2 \log^6(r))$, $l_2 = \mathcal{O}(r^2)$. Application of Algorithm 3.1 gives $PAQ \approx LU$, where P and Q are permutation matrices, and L and U are lower and upper triangular matrices, respectively. Then, the approximation error from the application of the sparse randomized LU decomposition is bounded by $\|LU - PAQ\|_F \leq \mathcal{O}(\Delta_r)$ with high probability.*

Proof. Choose $0 < \varepsilon < 1$ (ε affects the error of the decomposition) and $0 < \delta < 1$ (δ affects the probability that the decomposition is accurate). According to Algorithm 3.1, $\Omega_1 = \Pi_1 S_1 \in M_{k_1 \times n}$ where $\Pi_1 \in M_{k_1 \times l_1}$ is an SRFT matrix and S_1 is a random SEM. The pivoted LU decomposition of B is given by $PB = L_1 U_1$. Let $k_1 = \mathcal{O}(r \varepsilon^{-1} \log(r/\varepsilon))$ and $l_1 = \mathcal{O}(r^2 \log^6(r/\varepsilon) + r \varepsilon^{-1})$. Then from Corollary 3.2 it follows that $\|PA - L_1 L_1^\dagger PA\|_F < (1 + \varepsilon) \Delta_r$. Let

$$l_2 \geq \delta^{-1}(r^2 + r)/(2\varepsilon - \varepsilon^2)^2, k_2 \geq 4 \left[\sqrt{r} + \sqrt{8 \log(r l_2)} \right]^2 \log r.$$

Then, by Corollary 2.8, with high probability, $\Omega_2 L_1$ is left invertible. Thus,

$$\|L_1 L_1^\dagger PA - PA\|_F = \|L_1 (\Omega_2 L_1)^\dagger (\Omega_2 L_1) L_1^\dagger PA - PA\|_F.$$

Next, we bound $\|L_1 (\Omega_2 L_1)^\dagger (\Omega_2 L_1) L_1^\dagger PA - PA\|_F$ by the following:

$$\begin{aligned} \|L_1 (\Omega_2 L_1)^{-1} \Omega_2 PA - PA\|_F &= \|L_1 (\Omega_2 L_1)^{-1} \Omega_2 PA - L_1 (\Omega_2 L_1)^{-1} (\Omega_2 L_1) L_1^\dagger PA \\ &\quad + L_1 (\Omega_2 L_1)^{-1} (\Omega_2 L_1) L_1^\dagger PA - PA\|_F \\ &= \|L_1 (\Omega_2 L_1)^{-1} \Omega_2 (PA - L_1 L_1^\dagger PA) + L_1 L_1^\dagger PA - PA\|_F \\ &\leq \|L_1 (\Omega_2 L_1)^{-1} \Omega_2 (PA - L_1 L_1^\dagger PA)\|_F + \|L_1 L_1^\dagger PA - PA\|_F \\ &\leq \|L_1 (\Omega_2 L_1)^{-1} \Omega_2\|_2 \|PA - L_1 L_1^\dagger PA\|_F + \|L_1 L_1^\dagger PA - PA\|_F \\ &= (\|L_1 (\Omega_2 L_1)^{-1} \Omega_2\|_2 + 1) \|PA - L_1 L_1^\dagger PA\|_F. \end{aligned} \tag{4}$$

Let $L_1 = U \Sigma V^*$ be the SVD of L_1 , where $U \in M_{m \times k_1}$, $\Sigma \in M_{k_1 \times k_1}$, and $V \in M_{k_1 \times k_1}$. Then

$$\begin{aligned} \|L_1 (\Omega_2 L_1)^{-1} \Omega_2\|_2 &= \|U \Sigma V^* (\Omega_2 U \Sigma V^*)^{-1} \Omega_2\|_2 \\ &= \|U \Sigma V^* (\Sigma V^*)^{-1} (\Omega_2 U)^{-1} \Omega_2\|_2 \\ &= \|U (\Omega_2 U)^{-1} \Omega_2\|_2 \\ &= \|(\Omega_2 U)^{-1} \Omega_2\|_2 \\ &\leq \|(\Omega_2 U)^{-1}\|_2 \|\Omega_2\|_2. \end{aligned} \tag{5}$$

By combining Eqs. (4) and (5) with Corollary 2.8 and Theorem 2.4, we get

$$\begin{aligned} \|L_1 (\Omega_2 L_1)^{-1} \Omega_2 PA - PA\|_F &\leq \left(\frac{C(n, k_2)}{0.4(1-\varepsilon)} + 1 \right) \|PA - L_1 L_1^\dagger PA\|_F \\ &\leq 1.48(1 + \varepsilon) \left(\frac{C(n, k_2)}{0.4(1-\varepsilon)} + 1 \right) \Delta_r. \end{aligned}$$

From Algorithm 3.1, obtain

$$\|LU - PAQ\|_F = \|L_1 \tilde{L} U Q^* - PA\|_F = \|L_1 (\Omega_2 L_1)^{-1} \Omega_2 PA - PA\|_F \leq \mathcal{O}(\Delta_r),$$

which completes the proof. \square

3.1 Algorithm Complexity

Denote $m, n, r, k_1, k_2, l_1, l_2$ as in Algorithm 3.1. Assume, without loss of generality, that $m \geq n$. Then

1. Ω_1 construction takes $\mathcal{O}(n + l_1 k_1)$ operations.
2. $B = A \Omega_1^*$ computation takes $\mathcal{O}(mn + m l_1 \log(k_1))$ operations.
3. Computation of the pivoted LU decomposition of B takes $\mathcal{O}(m k_1^2)$ operations.

4. Ω_2 construction takes $\mathcal{O}(m + l_2 k_2)$ operations.
5. $\Omega_2 L_1$ and $(\Omega_2 L_1)^\dagger$ computation takes $\mathcal{O}(mk_1 + k_1 l_2 \log(k_2))$ and $\mathcal{O}(k_2 k_1^2)$ operations respectively.
6. $(\Omega_2 L_1)^\dagger \Omega_2 P A$ computation takes $\mathcal{O}(mn + nl_2 \log(k_2) + k_2 k_1 n)$ operations.
7. LU decomposition of $(\Omega_2 L_1)^\dagger \Omega_2 P A$ takes $\mathcal{O}(k_2^2 n)$ operations.
8. $L = L_1 \tilde{L}$ computation takes $\mathcal{O}(mk_1^2)$ operations.

This sums up to a total complexity of

$$\mathcal{O}(mn + mk_1^2 + nk_2^2 + ml_1 \log(k_1) + nl_2 \log(k_2) + k_1 l_2 \log(k_2)),$$

and the complexity of the decomposition of a sparse matrix A is

$$\mathcal{O}(\text{nnz}(A) + mk_1^2 + nk_2^2 + ml_1 \log(k_1) + nl_2 \log(k_2) + k_1 l_2 \log(k_2)).$$

4 Numerical Results

In this section, the performance of the algorithm is evaluated. The algorithm is implemented in MATLAB using complex matrices. The Sub-sampled Randomized Hadamard Transform (SRHT) [17] is used with real matrices instead of using the SRFT matrix to achieve an efficient computation.

4.1 Numerical rank growth

In this experiment, we consider a matrix of size $n = 5000$ where its numerical rank changes between 50 to 900, i.e., the first r singular values are 1 and the other are exponentially decaying from e^{-10} to e^{-200} . As shown in Figure 1, Algorithm 3.1 results in an approximation of the same order as the numerical rank, up to a small error.

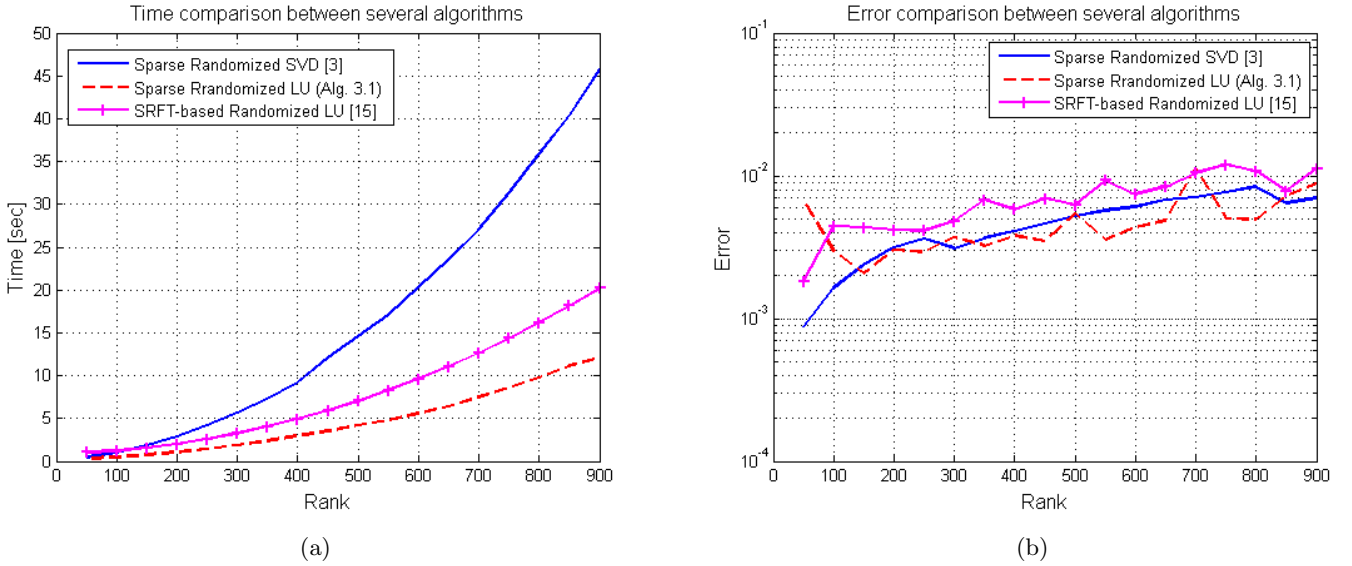
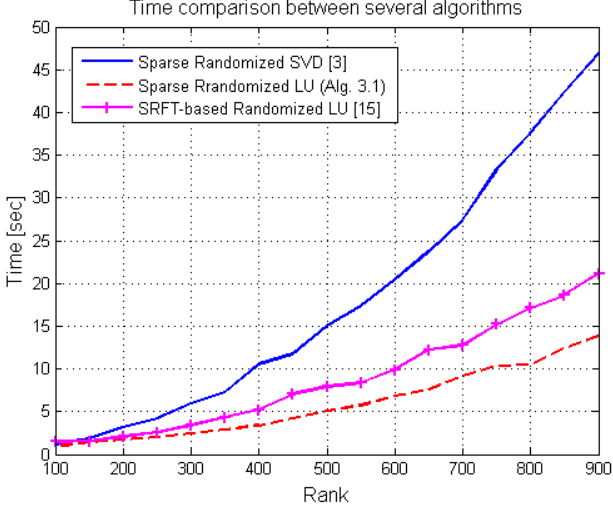


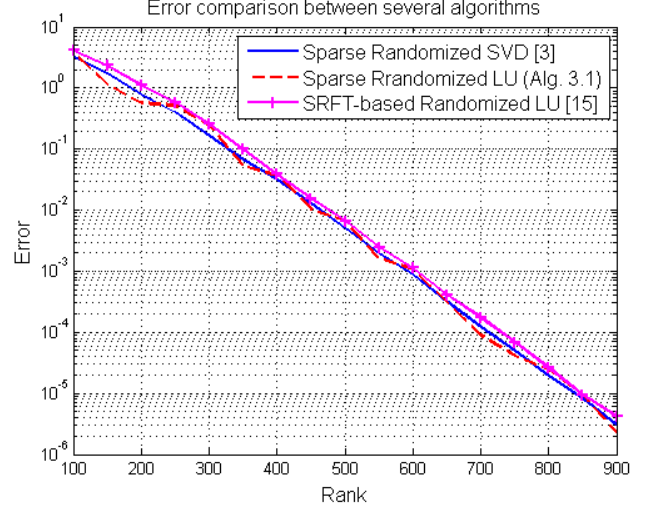
Figure 1: Results from the approximation of a matrix of size 5000×5000 with different numerical ranks. The numerical rank is shown on the x-axis. (a) the y-axis denotes the time each algorithm takes. (b) the y-axis denotes the error of each algorithm.

4.2 Improving the accuracy for a fixed matrix

In this experiment, we consider a matrix of size $n = 5000$ with singular values that decay exponentially from 1 to e^{-100} . We compute the r -th rank approximation by increasing r .



(a)



(b)

Figure 2: Results from the approximation a matrix of size 5000×5000 with exponentially decaying singular values. The approximation rank is shown on the x-axis. (a) the y-axis denotes the time each algorithm takes. (b) the y-axis denotes the error of each algorithm.

4.3 Running on GPU

The Sparse randomized LU decomposition (Algorithm 3.1) can be fully parallelized to run efficiently on a GPU card and on a distributed computing system such as Hadoop or Spark. In the following test, a 5000×5000 random matrix was processed in double precision on a GPU card using the MATLAB's GPU interface. MATLAB 2015a enables us to apply certain sparse matrices operations to the GPU. GTX Titan Black GPU card was used. Figure 3 compares the running time between GPU and CPU.

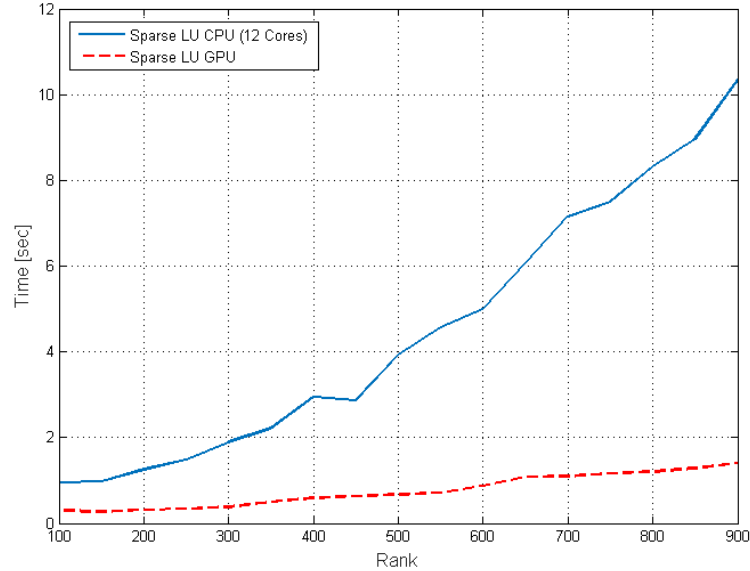


Figure 3: Running time on GPU vs. CPU of the randomized sparse LU decomposition (Alg. 3.1)

Conclusion

In this paper, the Sparse–Randomized–LU algorithm is presented. This algorithm utilizes sparse random projections that are combined with FFT–based projections for computing low rank LU matrix decompositions. The proposed technique was analyzed theoretically to achieve asymptotic bounds. The conducted numerical experiments compare the performance of the algorithm to other algorithms such as sparse SVD and fast randomized LU.

Acknowledgment

This research was partially supported by the Israeli Ministry of Science & Technology (Grants No. 3-9096, 3-10898), US-Israel Binational Science Foundation (BSF 2012282), Blavatnik Computer Science Research Fund and Blavatnik ICRC Funds.

References

- [1] D. ACHLIOPTAS AND F. MCSHERRY, *Fast computation of low-rank matrix approximations*, Journal of the ACM (JACM), 54 (2007), p. 9.
- [2] N. AILON AND B. CHAZELLE, *The fast Johnson–Lindenstrauss transform and approximate nearest neighbors*, SIAM J. Computing, 39 (2009), pp. 302–322.
- [3] K. L. CLARKSON AND D. P. WOODRUFF, *Low rank approximation and regression in input sparsity time*, in Proceedings of the 45th annual ACM symposium on Symposium on theory of computing, ACM, 2013, pp. 81–90.
- [4] P. DRINEAS, R. KANNAN, AND M. W. MAHONEY, *Fast monte carlo algorithms for matrices II: Computing a low-rank approximation to a matrix*, SIAM Journal on Computing, 36 (2006), pp. 158–183.
- [5] P. DRINEAS, M. W. MAHONEY, AND S. MUTHUKRISHNAN, *Relative-error CUR matrix decompositions*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 844–881.
- [6] G. H. GOLUB AND C. F. VAN LOAN, *Matrix computations*, vol. 4, John Hopkins University Press, 2012.
- [7] N. HALKO, P.-G. MARTINSSON, Y. SHKOLNISKY, AND M. TYGERT, *An algorithm for the principal component analysis of large data sets*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2580–2594.
- [8] N. HALKO, P.-G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288.
- [9] D. M. KANE AND J. NELSON, *Sparser Johnson–Lindenstrauss transforms*, Journal of the ACM (JACM), 61 (2014), p. 4.
- [10] P. MARTINSSON, V. ROKHLIN, AND M. TYGERT, *A randomized algorithm for the decomposition of matrices*, Applied and Computational Harmonic Analysis, 30 (2011), pp. 47–68.
- [11] J. NELSON AND H. L. NGUYÊN, *OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings*, in Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on, IEEE, 2013, pp. 117–126.
- [12] ———, *Sparsity lower bounds for dimensionality reducing maps*, in Proceedings of the forty-fifth annual ACM symposium on Theory of Computing, ACM, 2013, pp. 101–110.
- [13] ———, *Lower bounds for oblivious subspace embeddings*, in Automata, Languages, and Programming, Springer, Berlin Heidelberg, 2014, pp. 883–894.

- [14] M. RAAB AND A. STEGER, *“Balls into Bins” – A simple and tight analysis*, in Randomization and Approximation Techniques in Computer Science, Springer, 1998, pp. 159–170.
- [15] G. SHABAT, Y. SHMUELI, Y. AIZENBUD, AND A. AVERBUCH, *Randomized LU decomposition*, arXiv preprint arXiv:1310.7202, (2013).
- [16] A. SZLAM, Y. KLUGER, AND M. TYGERT, *An implementation of a randomized algorithm for principal component analysis*, arXiv preprint arXiv:1412.3510, (2014).
- [17] J. A. TROPP, *Improved analysis of the subsampled randomized Hadamard transform*, Advances in Adaptive Data Analysis, 3 (2011), pp. 115–126.
- [18] F. WOOLFE, E. LIBERTY, V. ROKHLIN, AND M. TYGERT, *A fast randomized algorithm for the approximation of matrixes*, Applied and Computational Harmonic Analysis, 25 (2008), pp. 335–366.